Agent Program for Providing Seamless Management to Web-based Monitoring System

T. Fukatsu, M. Hirafuji, and T. Kiura[1]

Abstract

In the agricultural field, one approach to solving long-term field monitoring is to employ a Web-based monitoring system known as “Field Servers” developed by Fukatsu and Hirafuji (2005). To use the system effectively, we developed an agent program which autonomously and flexibly manages the Field Servers. The main function of the program is to manage various Web-accessible items such as Field Servers, Network Cameras, Databases, and Web applications. It can operate only by installing the Java program in our computer and by adjusting some parameters. Throughout this paper we describe the function and details of the agent program so users can apply it to their situation. By focusing on the program configuration file (Profile) in which the conduct of the agent program is written, new users can perform more complex and intelligent operations easily. The agent program provides seamless management of a Web-based monitoring system.

Keywords: Field Server, Agent, Java, WWW, Monitoring

Introduction

The agent program (Fukatsu *et al.* 2006a) written in Java (JDK1.5.0), which has less machine dependence and provides easy network handling, can be easily operated on a java-installed computer. By applying to our laboratory and accepting the licensing agreement, it is possible to obtain this program via the Internet, E-mail, or CD-ROM. Although the program will run with only a program configuration file (Profile) as an argument, its function and application can be extended by preparing a meta-Profile list file (FS_List). These files are written in XML format, making them simple to edit.

In FS_List, there are blocks of target Profile nodes, named <Field_Server> and <NetCamera>. Each node has some parameters such as <Name> to identify nodes, <Profile> to indicate its file address with URL, <Status> to show its

[1] National Agricultural Research Center, Tsukuba, Ibaraki 305, Japan, fserver@zoushoku.narc.affrc.go.jp

status in autonomous agent mode, and so on (Fig. 1). Associated applications such as Data Viewer and MetBroker (Laurenson *et al.* 2002) refer to Profiles linked from the FS_List. When <Field_Server> and <NetCamera> are described in a single Profile as a package, they should be separated by a dummy Profile for these applications in order to work normally.

Processing Flow

The agent program starts with the following format:
java.exe –jar FS_Agent.jar (option flag) [Profile1 address] [Profile2 address] …
The option flags specify the mode of agent operation shown by Table 1. We can choose several flags if needed, and we can input some addresses of the target Profiles following option flags. The greater the number of Profiles added onto the program argument, the more this program performs multi-thread processing based on machine power to operate dozens of Profiles independently. This address can be set with either File form or URL form, so some standard forms on the Internet can be directly utilized.

The Profile consists of four parts, which are the basic parameter, object and site information, setting parameter of target data, and agent program instructions (Fig. 2). In the case of standard objects such as Field Servers, we can use the Profile form by simply changing some parts of the basic parameter such as object name, default IP address, interval time, and output home directory.

After the agent program creates processing threads corresponding to each Profile, each thread executes according to its Profile sequentially at regular time intervals set on the basic parameter, and sometimes the thread reloads it. When reloading, the thread searches, in order, for a valid Profile with an address list of reloading the next Profile (automatically adding the initial given Profile at the lowest priority) described in the basic parameter. By using this function, it can switch to the appropriate Profile according to the network situation (Fig. 3).

Profile Parameter

The basic parameter and the object and site information are shown as Fig. 4 and Fig. 5. The object and site information mainly describes database URL and deployed place, time, and condition. The agent program itself does not need them, but they are important for other useful applications, especially the database URL.

In the setting parameter of target data, there are two important functional parameters for extracting the desired value from raw html data and calculating equations with these values in which the agent program spooled. It can be

described as a packed node or two separate nodes (Fig. 6). The extracting method is, 1) search for the keyword (<Label> value, or <H_key> value if non-existent) and skipping it at the specified times in the <Key_loop> value from spooled data, 2) extract the enclosed value between the <H_key> and <T_key> value from the data after the keyword, 3) put the value into the variable defined by the node name.

The specified equation is calculated with the extracting value for output data or an indicator of action determination. The calculation format in the <Calculation> node describes the equation which is combined with numbers, arithmetic codes, and variables enclosed in $ marks. (In the old version of the agent program, the calculation format is only described in a linear expression with <Slope> and <Intercept> value.) By setting these parameters, the agent program flexibly treats various Web-based targets.

The agent program instructions, a main part of the Profile, instruct the thread of each Profile. The instructions are combined with action commands that emulate user operations on a Web browser. Some types of commands (Table 2) are described in <Action>. In the argument of these commands, we can use the representation such as variable enclosed in $ marks which replaces variable value, HHmmss which replaces current time (hour, minute, and second), and yyyMMdd which replaces current date (year, month, and day) and creates automatic directories.

These action commands are executed in descending order. Each command can be used with IF-THEN rules and Operator functions to determine whether the command should be executed or not according to the situation such as time, sensor value, and defined flags. The IF-THEN rules are described with <Node> which is set by “Date”, “Time” or variable name, <Value> which is set by fixed value or variable enclosed in $ marks, and <Operator>. The <Node> value is compared to the <Value> with the <Operator>, which is described with “gt” (greater than <Value>), “lt” (less than <Value>), or “eq” (equal to <Value>). It is possible in a single command to include two or more IF-THEN rules with Operators described with “and” or “or”. The program sequentially handles the previous result and next result of the IF-THEN rule with the Operator and makes the decision to execute the command based on the combined result of these rules (Fig. 7). Using this system, we can establish complicated condition judgments to control the system.

Data Handling

Data collected using the Xml command is normally recorded in <Save_Path>/yyyyMM/yyyyMMdd.xml, which is automatically created and

separated daily. In this XML data file, Date, Time and the variable value are recorded in <Data> created each cycle (Fig. 8). With the <Xsl_Name>, this file can be displayed in tabular form via the Web browser. When the XML data file is created, some assist files are also created in <Save_Path>. In <Name>_List.xml file, there is a list of the XML data files. <Name>.htm file provides a useful select menu of XML data files with <Name>_List.xml and <Name>_yyyyMM.xsl files created automatically. The Animation command also creates the html file for displaying animation, <Name>_List.xml file, <Name>.htm, and <Name>_yyyyMM.xsl files in the same way.

Stored data can be displayed plainly by these html/xml files and also graphically by Web applications such as “Field Server Data/Image Viewer” and “MetBroker”. Field Server Data/Image Viewer written in Java applet displays data with graphs and motion images at any range and period selected by GUI tool bars (Fig. 9). For example, we can see the animation extracted from the specified time of day. MetBroker is a middleware for weather data that provides agricultural models with consistent access to many different weather databases. MetBroker-compatible models and applications can be used by applying to the Field Servers’ database.

Applications

By combining these functions, this program can adapt to various situations. To build an emergency system at a local site’s agent program, it should use the function of <Xml_List>. By making a high priority Profile to the remote site via the Internet and a low priority Profile to the local site, the agent program operates the high priority Profile described as normal operation in a high-quality Internet connection. When the Internet is disconnected at a local site, it cannot reload the Profile so it reloads the low priority Profile described as emergency operation such as trying to re-connect the Internet, minimizing energy use, and so on.

Switching operations according to the monitoring data such as a surveillance system can be realized by using the Profile of Fig.10 with the Set and Agent commands. The main Profile thread operates periodically to monitor the reference value at short intervals. When it can’t access the value because of some emergency or the value exceeds the specified threshold by perceiving something, the switching operation is started independently by the Agent command which is made valid based on the value checked by the IF-THEN rule.

In the case of cooperating Web applications, the agent program sends parameters to the application using the Read or Post command (Fig. 11). In the type of http GET request, the parameters are described with the format of “URL?param1=value1&param2=value2&…” in the argument of the Read command.

We can use a variable value by enclosing the input item in $ marks in the argument. In the type of POST request, we can set the parameters in the spooled data or the argument. After calculating in the Web application, we can get the result in the spooled data. This makes it possible to distribute calculation tasks on the Internet.

Future Work

The agent program can operate flexibly with the proper Profile. To make our system user friendly, we are developing GUI tools for making Profiles and managing the program (Fukatsu *et al.* 2006b). We are developing various Web applications to analyze data and images to expand the functionality of our system. We are also upgrading the agent program by adding various functions such as including other Profiles as acceptable formats, allowing synchronous cooperation among threads, avoiding agent access collision, allowing other protocols (FTP, Mail, SQL) to access our system, and developing autonomous operation. By improving the implementation of the program, we will manage more targets effectively with small resources and efforts.

References

Fukatsu, T., M. Hirafuji 2005. Field Monitoring Using Sensor-Nodes with a Web Server. Journal of Robotics and Mechatronics, 17(2):164-172.

Fukatsu, T., M. Hirafuji, and T. Kiura 2006a. An Agent System for Operating Web-based Sensor Nodes via the Internet. Journal of Robotics and Mechatronics, 18(2):186-194.

Fukatsu, T., M. Hirafuji, and T.Kiura 2006b. A Distributed Agent System for Managing a Web-based Sensor Network with Field Servers. Proc. of the 4th World Congress on Computers in Agriculture: 223-228.

M. R. Laurenson, T. Kiura, and S. Ninomiya 2002. Providing Agricultural Models with Mediated Access to Heterogeneous Weather Databases. Applied Engineering in Agriculture, 18(5): 617-625.

Tables

Table 1. List of the option flags.

-c	Operates on the Profiles continuously as soon as a series of cycle has been finished, regardless of the interval time set on them.
-l	Outputs more detailed log information about the operating conditions.
-w**	Starts each Profile at the specified milliseconds interval in order to disperse computer task and network access (= -w5000).
-d**	Checks Profiles regularly in the specified directory and automatically starts or stops them.

Table 2. Action commands of the agent program instructions.

Read (<addr>) (<key>): Access the specified address (default: <IP>), which can be formed by a file or URL, and spool its data. The <key> formed in "username:password" is sent in some Web pages with basic authentication.
Post (<URL>) (<data>): Access the specified URL with POST format, and spool the received data. The POST data can select the specified data or spooled data (in default).
Analyze: Extract and calculate data from spooled data with the setting parameter of target data.
Write <file>: Write the spooled data to the specified file address, which can be described with the relative path from <Save_Path>.
Xml (<file>): Write all variable values to the specified file in XML format with the append mode (default: <Save_Path>/yyyyMM/yyyyMMdd.xml).
Htm (<file>): Write all variable values to the specified file in HTML format with the overwrite mode (default: <Save_Path>/<Name>_now.htm).
Wait <msec>: Wait the specified number of milliseconds.
Set <name> (<value>): Set directly the specified value into the specified variable created if there is a new one. Default value is inputted by the status flag which indicates whether before command finished normally or not.
Agent <Profile> (<status>): Create a new thread with the specified Profile and operate it independently in default status. When "stop" is set into the <status>, the thread with the specified Profile is stopped.
Execute <command> (<async>): Execute the specified command in a System process. This command expands the operation of the agent program. By setting <async> to "on", one can cause the next action command starts without waiting for the end of this command execution.
Thumbnail <scale>: Zoom the spooled image data to the specified percentage scale. This function can also be realized by accessing a Web application that has a comparable function. It expands the agent function without changing the agent program by simply making or finding Web applications that execute the desired function.
Animation <dir> (<link_dir>): Make an HTML file in which all image files in the specified directory are displayed with serial image data like animation. When the <link_dir> is in the thumbnailed image files, this page is displayed with the thumbnailed animation for reducing capacity, while the original image data can be easily accessed by clicking the thumbnailed image data which is linked to it. A user can inspect a moving image with only a few network loads and view important images with high resolution.

Figures

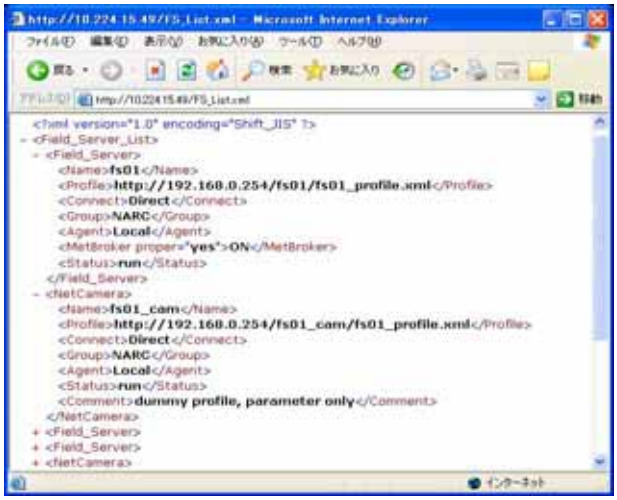


Fig.1 FS_List.xml.

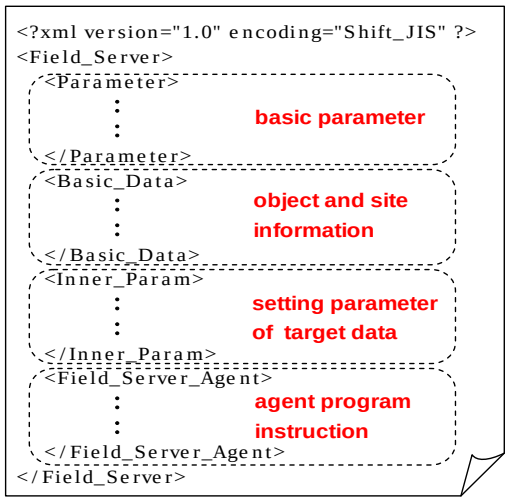


Fig.2 Profile.xml.

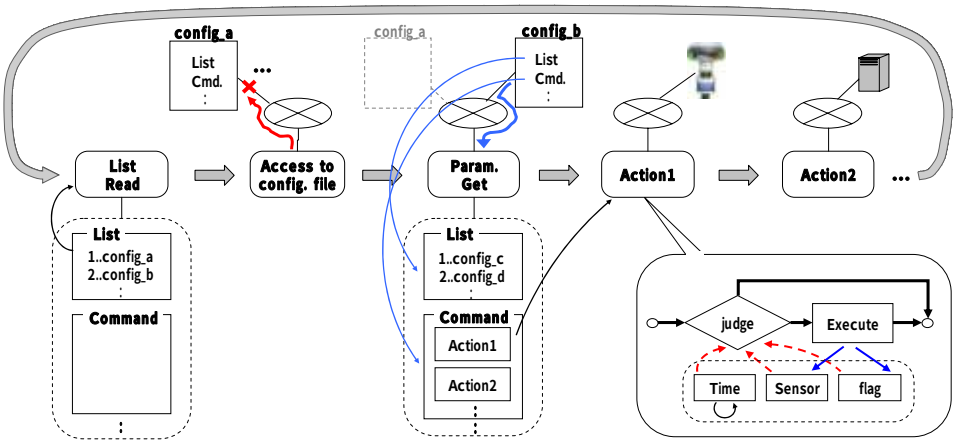


Fig.3 Agent program flow.

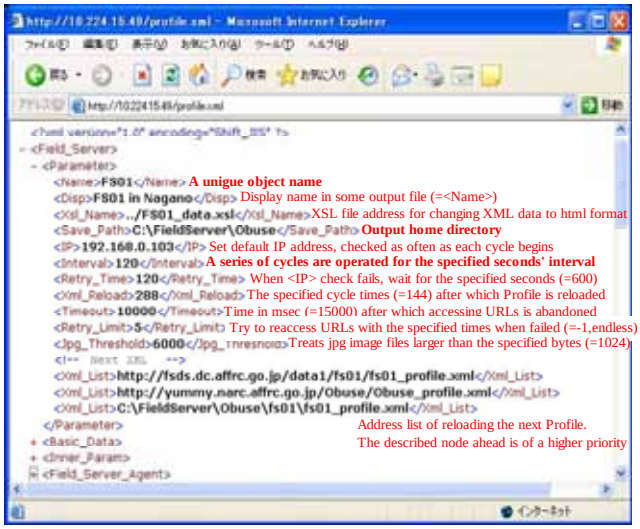


Fig.4 Basic parameter.

Fig.5 Object and site information.

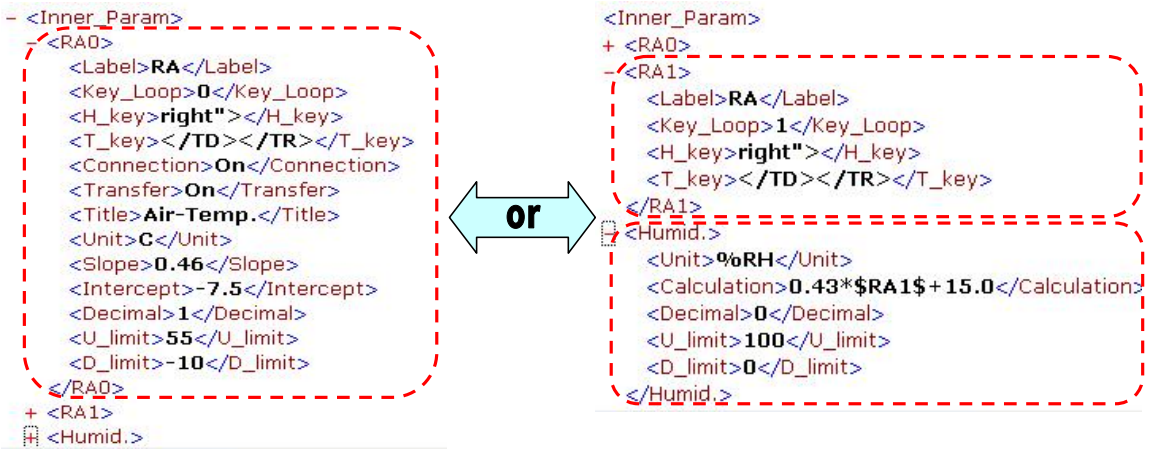


Fig.6 Setting parameter of target data.

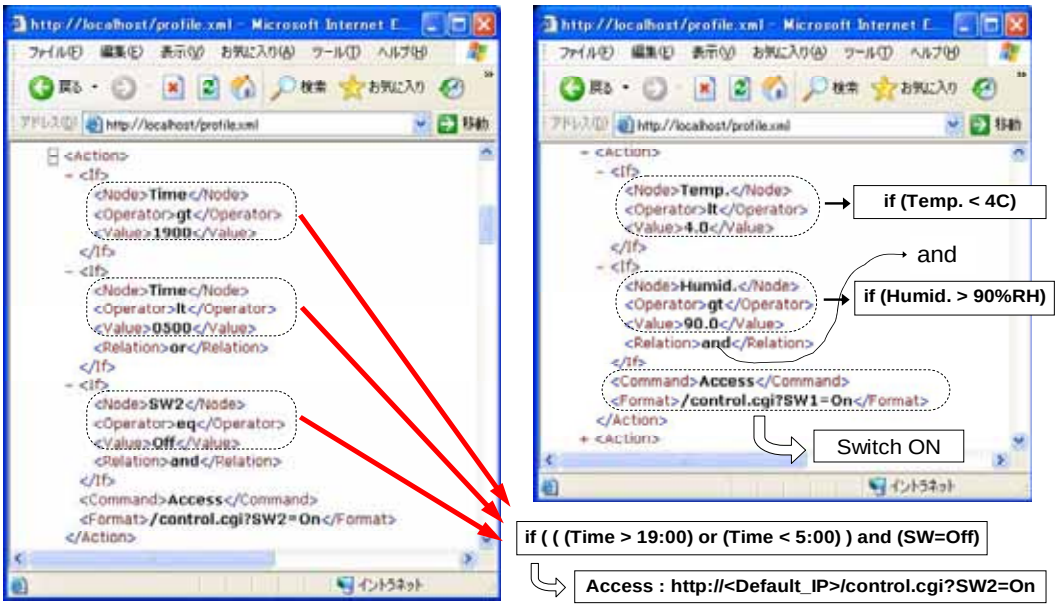


Fig.7 IF-THEN rules and Operator functions.

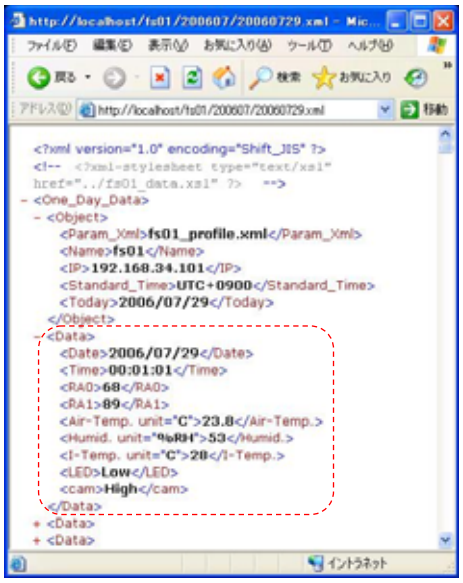


Fig.8 XML data file.

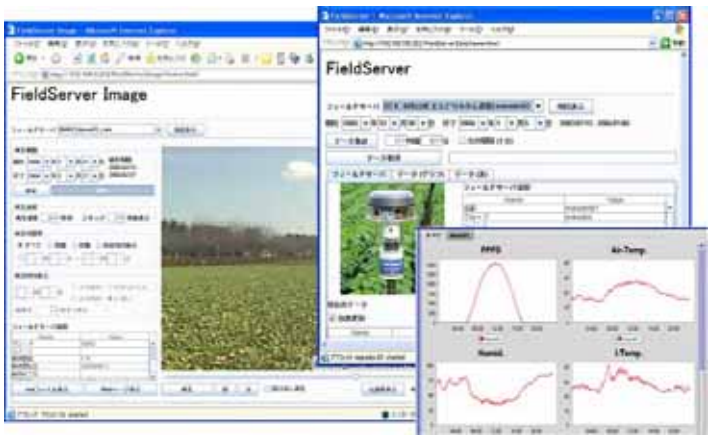
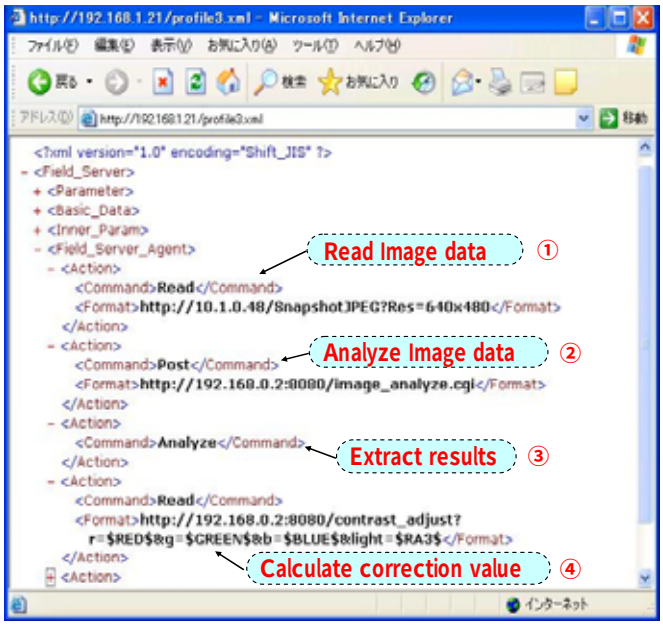


Fig.9 Field Server Data/Image Viewer.

Fig.10 Sample Profile of surveillance application.



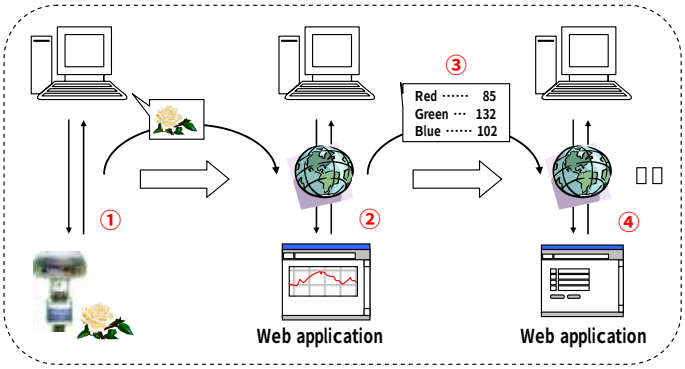


Fig.11 Sample Profile of cooperating Web application.